micro sec 6

loop

DJNZ Reg, lablel

بعمل decrement للـ reg وبـ check كل

شوية ، طول ما هو مش صفر بعمل jump للـ label

ex

Reg = value

DJNZ Reg, loop

بعمل decrement للـ reg

طول ما هو ليس بصفر بعمل loop

- أكبر قيمة للـ reg هيا FF (255)

لو عندي loop تحتاج قيمة أكبر من 255 هعمل أكثر من loop جوه بعض nested loops

- لو عندي فعلاً 700 . هعمل 2 loops

واحده 10 والتانية 70 ، والناتج يكون حاصل ضربهم .

Jump

↳ Conditional Jump

لازم يعمل (operation) ثم أعمل (check flag) على
أساسه بعرف أخد أى (action) فى البرنامج.

ex: JZ lablel

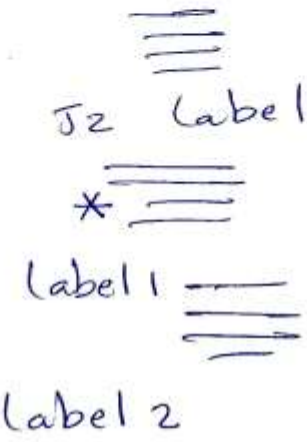
← يعمل (check) على (zero flag) وحتى
Jump لـ label .

if equal to zero Jump to label

else if not equal to zero ---

(unconditional Jump) يعمل (*) فى الكتابة

لـ (label 2)



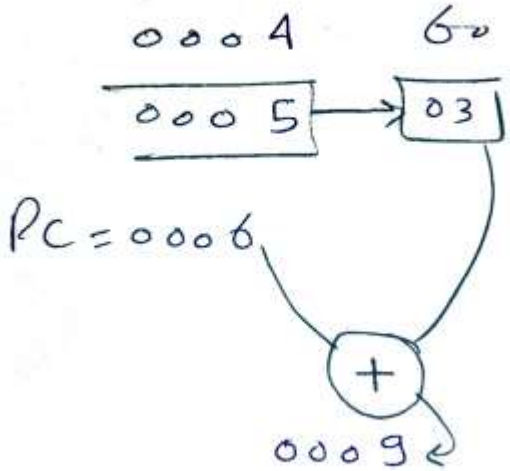
SJMP label →

تقدر تعمل (Jump) فوحدة تحت بحد أقصى (128 bit)

LJMP label (long Jump)

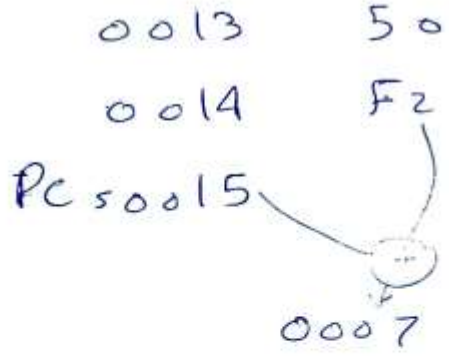
يعمل (Jump) لأى مكان فى البرنامج.

لأنه (3byte) واحدة لـ (opcode) واتنين لـ (address)



look at Page 102

at JZ Next



Call

(subroutine) لما بكتب شوية أكواد وفي النهاية عايز أعمل

فبعمله Call

لا بتعمل (Jump) بتكمل الكود بتاعك

لكنه مش بترجع للبداية .

أما في ال (Call) بتستخدم

أمر (return) ترجع بيه

للبداية.

Call sub 1

sub 1:

RET return

- من امثلة العمليات لامر (Call) انه يعمل Push لـ (PC) ويعمل تغيير لـ (PC) لـ (sub1) عشان يشاور ليه.

```
PUSH PC
change Pc to sub1
sub1: ═══════
      ═══════
      ═══════
RET          =>PoP PC
```

- قبل الجزء بتاع (sub1) هضع (HLT) معناها نهاية البرنامج.

Long Call (LCall)

يعمل (Call) فى اى مساحة فى البرنامج

Absolute Call (ACall)

- يعمل (Call) فى حدود (2K-byte) فقط.

- مثال فى صفحة 117.

← لو فى ال (main) وال (subroutine) بتستعمل نفس ال (registers) ال (Call) مش دوره انه يحمى ال (register) لأنه بيتعامل مع ال (Push) وال (return) فقط.

← فى الحاله دى قبل بداية ال (subroutine) هعمل (Push) للقيم بتاعة ال (register) اللى هستخدمها وفى الآخر هعمل (PoP) ليهم بس بالعكس.

← لو عملت (Push) وفى الآخر معملتش (PoP) مش هيطلع قيم ال (Pc) المطلوبة.

← لازم كل (Push) فى البداية يقابلها نفس ال (PoP)

ال (inst.) يتم تعريفه بال (Crystal oscillator) و (Machine cycle)

هو (Cycle) يتم تنفيذ ال inst. فيها.

لـ ال (Microcontroller) بداخله مجموعة من ال (operations) كل (operation) بتحتاج كام (Cycle)

→ # of clocks.

← ال (machine cycle) لل (atmel) عبارة عن (12 Clocks).

→ look at Page 122.

لم يستخدم الموضوع ده في اي (delay)
لم المثال في صفحة 123.

LCD

مثال للإنك ← تستخدم (HW) جواها (MC).

عبارة عن byte
← تبعتله (data) تتعرض عليه و (commands) يعرض بيها ال (data).

← ليها 14 Pin

14-Pin
- → 8 - data
- → 3 :
  - V_{CC} supply
  - V_{BB} Contrast
  - V_{DP} ground
- → 3 :
  - E → enable.
  - R/W
  - Rs → 0 Command, → 1 data

6

ال (LCD) تشتغل إزاي؟

عشان تتعامل مع ال (LCD)

- تبعت ليها (data) و (Commands) / أنشغل بيها ال (data)
- تبعت ليها (Commands) دبيني وبينها (Protocol) نفهم بيه ال (Commands) دي.

R/W̄ → 0 write
→ 1 read.

— لو بعت حاجة من ال (μc) لل (LCD) وبقوله يقرأها الموضوع ده بيحصل بال E (enable).

كل الحروف اللي إنتا بتبعته عشان يتعرض بيتخزن في (Memory)

LCD Memory

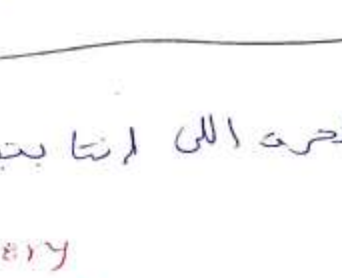
→ DDRAM ,

— كل (character) بتشوفه لازم يكون ليه مقابل في ال (memory)

~~يعني لو عرضت حاجة في المكان 3~~

ex show 'A'
row 1 Col 3

DDRAM



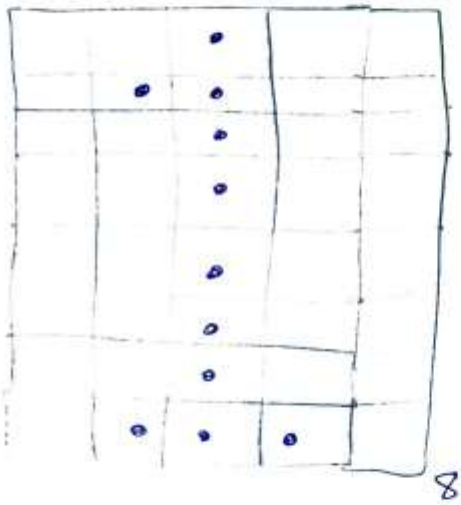
المكتوب هو الكود بتاع حرف ال A.

بس هو لازم يعرف حرف A تترسم إزاي ودي وظيفة (CGROM)

→ CGROM

← تقرى كل (Ascii code) بيترسم (زاى) بيكون متخزن فيها شكل رسمة الحروف .

← شكل حرف "1".



CGRAM

← هتوله اخزلك قيمة بس أنا اللى هعلى ال (Array) بتغير .

لو عايز أعمل (create) لحرف جديد مش موجود ← (Special character).

ال task

← هتور

لو (Calculator) بتعمل الأربع عمليات جمع وطرح وضرب وقسمة . على ال (LCD)